New explicit iteration algorithms for solving coupled continuous Markovian jump Lyapunov matrix equations

Zhaolu Tian[a,*], C.M. Fan[b], Youjun Deng[c], P.H. Wen[d]

[a] *College of Data Science, Taiyuan University of Technology, Taiyuan 030024, P.R.China*

[b] *Department of Harbor and River Engineering & Computation and Simulation Center, National Taiwan Ocean University, Keelung 20224, Taiwa*

[c] *School of Mathematics and Statistics, Central South University, Changsha 410083, P.R.China*

[d] *School of Engineering and Materials Science, Queen Mary University of London, London E14NS, UK*

Abstract

In this paper, based on the Smith iteration [30], an inner-outer (IO) iteration algorithm for solving the coupled Lyapunov matrix equations (CLMEs) is presented. First, the IO iteration algorithm for solving the Sylvester matrix equation is proposed, and its convergence is analyzed in detail. Second, the IO iteration algorithm for solving the CLMEs is constructed. By utilizing the latest estimation, a current-estimation-based and two weighted IO iteration algorithms are also given for solving the CLMEs, respectively. Convergence analyses indicate that the iteration solutions generated by these algorithms always converge to the unique solutions to the CLMEs for any initial conditions. Finally, Several numerical examples are provided to show the superiority of the proposed numerical algorithms.

Key words: *Smith iteration; Inner-outer iteration; Convergence; Lyapunov matrix equation*

1. Introduction

The CLMEs are very important in analysis and design for continuous-time Markovian jump linear systems, which can be used to model dynamic systems related to abrupt changes in their structures and parameters due to component failure or repairs, environment changes and changing subsystem interconnections. This kind of systems have a wide range of applications, such as network control systems [5] and fault tolerant control systems [7], etc. In [2], the stochastic controllability and stabilizability have been investigated for continuous-time Markovian jump linear systems. The moment stability of this kind of systems was studied in [32]. In [2,5], it was showed that both the stochastic and moment stability of the Markovian jump linear system can be characterized by the existence of unique positive definite solutions to the CLMEs.

The aforementioned facts indicate that the CLMEs play an important role in stability analysis and stabilizing controller design. Therefore, many approaches have been presented for solving

*Corresponding author.
E-mail address: tianzhaolu2004@126.com (Z. Tian).

the CLMEs related to the continuous-time Markovian jump systems during the last decades. Based on the Kronecker products, the exact solutions to the coupled matrix equations were explicitly obtained in [9], but this algorithm needed excessive computer storage and was prohibitive for large scale systems. In order to overcome this problem efficiently, iteration algorithms have been presented in the literatures for matrix equations [6,17,20,23]. In [12], a parallel iteration algorithm was proposed for solving the CLMEs, in which some independent standard continuous Lyapunov matrix equations needed to be solved in each iteration step. For high dimension matrices, the solution to the standard Lyapunov equation is still costly, and thus the algorithm in [12] was an implicit iteration algorithm. By introducing the latest estimation, a modified preceding implicit iteration algorithm was developed in [8]. A weighted implicit iteration algorithm in [16] was given for solving the coupled Lyapunov matrix equations, and its convergence rate was improved significantly if the weighting parameters were appropriately chosen. Some explicit iteration algorithms, for example, the gradient-based iteration algorithms [22,26,28], have been proposed for solving the CLMEs. Several reduced-rank gradient-based iteration algorithms in [29] were constructed for solving the generalized coupled Sylvester matrix equations, which can also be used to solve the CLMEs. Recently, some efficient explicit iteration algorithms have been developed, such as [24,27]. In [15], the iteration algorithms based on positive operator were presented for solving Itô stochastic systems with Markovian jumps, which convergence was obtained according to the properties of some positive operators associated with the stochastic system. In [18,21], by using the ST technique [34] to decompose the stochastic systems with Markovian jumps into N decoupled linear subsystems, some policy iteration algorithms have been established for solving the stochastic coupled algebraic Riccatic equation for the continuous-time Itô stochastic systems with Markovian jumps. In [11], an explicit iteration algorithm was proposed for solving the CLMEs. By introducing the latest estimation, a current-estimation-based and an accelerated Smith iteration algorithms were also given, respectively.

In this paper, we present an IO iteration algorithm for solving the CLMEs associated with the continuous-time Markovian jump linear systems. First, we propose the IO iteration algorithm for solving the Sylvester matrix equation, and prove its overall convergence. Next, we use the IO iteration algorithm to solve the CLMEs. In order to improve the convergence rate of the IO iteration algorithm, a current-estimation-based IO iteration algorithm is constructed by utilizing the latest estimation, and it is proved that the algorithm can monotonically converge to the unique positive definite solutions to the corresponding matrix equations with zero initial conditions. Moreover, a necessary and sufficient condition is given for the proposed algorithm with any initial conditions to be convergent. Furthermore, two weighted IO iteration algorithms are also presented. Finally, several numerical examples are implemented to demonstrate the effectiveness of the proposed iteration algorithms.

Throughout this paper, for a matrix $A \in R^{n\times n}$, A^T and $\rho(A)$ denote its transpose and spectral radius, respectively. For two integers m and n with $m \leq n$, $\Pi[m,n]$ denotes the set $\{m, m+1, \cdots, n\}$. For a matrix $A = [a_1\ a_2\ \cdots\ a_n]$, $\mathrm{vec}(A) = [a_1^T\ a_2^T\ \cdots\ a_n^T]^T$. The notation $A \otimes B$ represents the Kronecker products of the matrices A and B. The matrix $E > 0$ means that E is real symmetric and positive definite. The matrix tuple $\mathcal{F} = \{F_1, F_2, \cdots F_n\} > 0$

implies that all the matrices $F_i > 0,\ i \in \Pi[1,n]$. For two matrices A and B, we define $\lambda(A,B) = \{\tau|\det(\tau\mathrm{A}-\mathrm{B})=0\}$. For a complex number a, $\mathrm{Re}(a)$ is its real part. In what follows, it should be stated that the sum is zero if the upper limit of the sum notation is less than the lower limit.

2. Previous results

Consider the following continuous-time Markovian jump linear system

$$\dot{x} = A_{r(t)}x(t), \tag{2.1}$$

where $x(t) \in R^n$ is the state vector, $r(t)$ is a continuous-time Markovian random process, which takes values in a discrete finite set $\Omega = \{1,2,\cdots,N\}$, and $A_{r(t)} \in R^{n\times n}$ is the mode-dependent system matrix. The dynamics of the probability distribution of the Markov chain is described by the following differential equation

$$\dot{\pi}(t) = \pi(t)P, \tag{2.2}$$

where π is an N-dimensional row vector of unconditional probabilities and P is the transition rate matrix denoted by $[p_{ij}]_{n\times n}$, which elements satisfy $p_{ii} < 0,\ p_{ij} \geq 0 (i \neq j)$ and $\sum\limits_{j=1}^{n} p_{ij} = 0$.

Let $x(0) = x_0$ and $r(0) = r_0$, the definition of stochastic stability for (2.1)-(2.2) can be described as follows:

Definition 2.1 [1]. The continuous-time Markovian jump linear system (2.1)-(2.2) is stochastically stable if for any $x_0 \in R^n$ and $r_0 \in \Omega$, there holds

$$E\left\{\int_0^\infty \|x(t)\|^2 |x_0, r_0\right\} < \infty.$$

The CLMEs for the preceding Markovian jump linear system (2.1)-(2.2) have the following form:

$$A_i^T K_i + K_i A_i + \sum_{j=1}^{N} p_{ij}K_j + Q_i = 0,\ Q_i > 0,\ i \in \Omega, \tag{2.3}$$

where $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$ is the unknown matrix tuple, and $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_N\} > 0$ is an arbitrarily given positive definite matrix tuple. In fact, the CLMEs (2.3) are a special case of (21) [15] with $r = 0$.

It is well known that the CLMEs (2.3) can be rewritten as follows:

$$\mathcal{A}_i^T K_i + K_i \mathcal{A}_i + \sum_{j=1, j\neq i}^{N} p_{ij}K_j + Q_i = 0,\ Q_i > 0,\ i \in \Omega, \tag{2.4}$$

where $\mathcal{A}_i = A_i + \frac{p_{ii}}{2} I\ (i \in \Omega)$, and I is an identity matrix with appropriate dimension.

Lemma 2.1 [1]. The continuous-time Markovian jump linear system (2.1)-(2.2) is stochastically stable if and only if the CLMEs (2.3) have a unique solution $\mathcal{K} > 0$ for any $\mathcal{Q} > 0$.

Lemma 2.2 [11]. If the Markovian jump linear system (2.1)-(2.2) is stochastically stable, then the matrix $\mathcal{A}_i$ $(i \in \Omega)$ in (2.4) is Hurwitz stable, which means that all the eigenvalues of $\mathcal{A}_i$ $(i \in \Omega)$ lie on the left-hand plane.

Many iteration algorithms have been proposed for solving the CLMEs (2.3) and (2.4), some of them are listed as follows.

Lemma 2.3 [8]. If the Markovian jump linear system (2.1)-(2.2) is stochastically stable, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ generated by the following iteration algorithm

$$\mathcal{A}_i^T K_i(m+1) + K_i(m+1)\mathcal{A}_i = -\sum_{j=1}^{i-1} p_{ij} K_j(m+1) - \sum_{j=i+1}^{N} p_{ij} K_j(m) - Q_i,\ i \in \Omega \tag{2.5}$$

converges to the unique solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$ to the CLMEs (2.4) for zero initial conditions.

Lemma 2.4 [19]. Assume that the CLMEs (2.4) have a unique solution, if $|\zeta| < 1$ for any $\zeta \in \lambda(\bar{H}, \bar{G})$, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ derived from the following iteration algorithm

$$\begin{aligned} &\mathcal{A}_i^T K_i(m+1) + K_i(m+1)\mathcal{A}_i \\ &= -\sum_{j=1}^{i-1} p_{ij}\Big((1-\gamma)K_j(m+1) + \gamma K_j(m)\Big) - \sum_{j=i+1}^{N} p_{ij} K_j(m) - Q_i,\ i \in \Omega \end{aligned} \tag{2.6}$$

converges to the unique solution $\mathcal{K}$ to the CLMEs (2.4) for any initial conditions, where $0 \le \gamma < 1$ is a tunable parameter, and the matrices $\bar{H},\ \bar{G}$ are defined as in Theorem 2 [19].

From Algorithms (2.5) and (2.6), it is clear that N standard continuous Lyapunov matrix equations need to be solved in each iteration step, so Algorithms (2.5), (2.6) are also called the implicit iteration algorithms due to this. In order to avoid solving the Lyapunov matrix equations, some explicit iteration algorithms have been constructed in [11,22].

Lemma 2.5 [22]. If the CLMEs (2.3) have a unique solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\} > 0$, consider the following iteration algorithm

$$K_i(m+1) = K_i(m) - \mu\left(A_i\Delta_i(K) + \Delta_i(K)A_i^T + \sum_{j=1}^{N} p_{ji}\Delta_j(K)\right),\ i \in \Omega, \tag{2.7}$$

where μ is the step size and

$$\Delta_i(K) = A_i^T K_i(m) + K_i(m)A_i + \sum_{j=1}^{N} p_{ij} K_j(m) + Q_i,\ i \in \Omega.$$

Then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ generated by (2.7) converges to the unique solution $\mathcal{K}$ to the CLMEs (2.3) for any initial conditions if and only if

$$0 < \mu < (2/\|\Xi\|_2^2),$$

where Ξ is the matrix defined as in Theorem 4 [22].

Lemma 2.6 [11]. Assume that the Markovian jump linear system (2.1)-(2.2) is stochastically stable, consider the following iteration algorithm

$$\begin{aligned} &K_i(m+1) \\ &= U_i^T K_i(m) U_i + 2qD_i^T \left(-\sum_{j=1}^{i-1} p_{ij} K_j(m+1) - \sum_{j=i+1}^{N} p_{ij} K_j(m) - Q_i \right) D_i,\ i \in \Omega, \end{aligned} \tag{2.8}$$

where q is a tunable parameter, and the matrices U_i, D_i are defined as in Theorem 1 [11]. If $q < 0$, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ obtained from (2.8) with zero initial conditions converges to the unique solution $\mathcal{K}$ to the CLMEs (2.4).

In [11], an accelerated Smith iteration algorithm and corresponding convergence theorem are also given. The algorithm is expressed as follows:

$$\begin{aligned} K_i(m+1) \ &= U_i^T K_i(m) U_i + 2D_i^T \Bigg(-\sum_{j=1}^{i-1} p_{ij} \big((q-1) K_j(m+1) + K_j(m) \big) \\ &\quad -q \sum_{j=i+1}^{N} p_{ij} K_j(m) - qQ_i \Bigg) D_i,\ i \in \Omega. \end{aligned} \tag{2.9}$$

3. An IO iteration algorithm for solving $AX + XB = C$

In [4], an IO iteration algorithm has been presented for solving the following linear system to obtain PageRank vector [25,33]:

$$(I - \eta P)x = (1 - \eta)v \tag{3.1}$$

with $0 < \eta < 1$, and P is a column-stochastic matrix. Based on the following matrix splitting

$$I - \eta P = (I - \beta P) - (\eta - \beta)P, \tag{3.2}$$

then the iteration sequence for solving (3.1) can be expressed by

$$(I - \beta P)x_{k+1} = (\eta - \beta)Px_k + (1 - \eta)v, \quad k = 0, 1, \cdots, \tag{3.3}$$

where $0 < \beta < \eta$. The iteration sequence (3.3) is the so-called outer iteration.

In order to solve the linear system efficiently with the coefficient matrix $I - \beta P$ in each iteration of (3.3), an inner Richardson iteration is used to approximate x_{k+1}. Setting the right-hand side of (3.3) as

$$f = (\eta - \beta)Px_k + (1 - \eta)v,$$

and defining the following inner linear system

$$(I - \beta P)y = f, \tag{3.4}$$

then (3.4) can be solved by the inner iteration

$$y_{j+1} = \beta P y_j + f,\ j = 0, 1, 2, \cdots, l-1, \tag{3.5}$$

where y_0 is given by x_k as the initial guess and y_l is treated as the new x_{k+1}.

Next, we will apply the IO iterations (3.3) and (3.5) to solve the following Sylvester matrix equation

$$AX + XB = C, \tag{3.6}$$

where $A \in R^{m\times m},\ B \in R^{n\times n},\ C \in R^{m\times n}$ are known matrices, and $X \in R^{m\times n}$ is the unknown matrix to be determined. When $A = B^T$ and $C = C^T$, Eq. (3.6) becomes the well-known Lyapunov matrix equation. Eq. (3.6) admits a unique solution if and only if A and $-B$ possess no common eigenvalues [14].

By using the principle of the Smith method in [11,35], Eq. (3.6) can be transformed into the following equivalent form:

$$(\psi I_m - A)X(\psi I_n - B) - (\psi I_m + A)X(\psi I_n + B) = -2\psi C, \tag{3.7}$$

where ψ is a tunable parameter, I_m and I_n are the $m\times m$ and $n\times n$ identity matrices, respectively.

Premultiply (3.7) by $(\psi I_m - A)^{-1}$ and postmultiply (3.7) by $(\psi I_n - B)^{-1}$, then we have

$$X - EXF = Q \tag{3.8}$$

with

$$\begin{cases} E = (\psi I_m - A)^{-1}(\psi I_m + A), \\ F = (\psi I_n + B)(\psi I_n - B)^{-1}, \\ Q = -2\psi(\psi I_m - A)^{-1}C(\psi I_n - B)^{-1}. \end{cases}$$

According to the properties of the Kronecker products [3] and Eq. (3.8), we obtain the following linear system

$$(I - F^T \otimes E)\mathrm{vec}(X) = \mathrm{vec}(Q).$$

By using the IO iterations (3.3), (3.5) and the following matrix splitting

$$I - F^T \otimes E = (I - \alpha F^T \otimes E) - (1-\alpha)F^T \otimes E,$$

we obtain the following stationary iteration sequence for solving Eq. (3.8):

$$(I - \alpha F^T \otimes E)\mathrm{vec}(X_{k+1}) = (1-\alpha)(F^T \otimes E)\mathrm{vec}(X_k) + \mathrm{vec}(Q),\ k = 0, 1, 2, \cdots$$

with $0 < \alpha < 1$, which is equivalent to

$$X_{k+1} - \alpha E X_{k+1} F = (1-\alpha)EX_kF + Q,\ k = 0, 1, 2, \cdots. \tag{3.9}$$

In what follows, we regard (3.9) as the outer iteration.

Let $W = (1-\alpha)EX_kF + Q$ and $Y = X_{k+1}$, then it follows from (3.9) that

$$Y - \alpha EYF = W,$$

and X_{k+1} can be computed by using an inner iteration as follows:

$$Y_{j+1} = \alpha EY_jF + W,\ j = 0, 1, 2, \cdots, m_k - 1, \tag{3.10}$$

where $Y_0 = X_k$ as the initial guess, and assign Y_{m_k} as the approximate solution to X_{k+1} in (3.9).

Algorithm 1: The IO iteration algorithm for solving Eq. (3.8)

Input: $E,\ F,\ Q,\ \alpha,\ \varrho$

Output: X

1:$X \leftarrow Q$

2: $Y \leftarrow EXF$

3: **while** $\|Q + Y - X\|_F \geq \varrho$

4: $W \leftarrow (1-\alpha)Y + Q$

5: **for** i=1:m_k

6: $X \leftarrow \alpha Y + W$

7: $Y \leftarrow EXF$

8: **end**

9: **end while**

In Algorithm 1, lines 1 and 2 initialize $X = C$ and $Y = EXF$. W in (3.10) is computed in line 4. The inner iteration defined in (3.10) is implemented from line 5 to line 8. The matrix $Y_{j+1} - \alpha E Y_{j+1} F$ is given by $X - \alpha Y$, since now X holds Y_{j+1} and Y is computed by EXF in line 7 as well as in line 2. Upon exit from the algorithm, X is the desired approximation of the exact solution to Eq. (3.8).

3.1 Convergence analysis of the IO iteration algorithm

First, we rewrite the IO iteration algorithm as the following two-stage matrix splitting iteration framework:

$$\begin{cases} X_{k,0} = X_k, X_0 = Q,\ X_{k+1} = X_{k,m_k}, \\ X_{k,j+1} = \alpha E X_{k,j} F + (1-\alpha) E X_k F + Q,\ k = 0, 1, 2, \cdots, \\ j = 0, 1, 2, \cdots, m_k - 1. \end{cases} \tag{3.11}$$

Theorem 3.1. Let $\psi > 0$ and $0 < \alpha < 1$, and m_k be the number of the inner iteration steps at the $k-$th outer iteration. If A and B are stable, then the iteration sequence $\{X_k\}_{k=0}^{\infty}$ derived from (3.11) converges to the exact solution X^* to Eq. (3.8). Moreover, the IO iteration algorithm converges faster than the Smith method.

Proof. If A and B are stable, then $\mathrm{Re}(\tilde{\psi}_i) < 0$ $(i = 1, 2, \cdots, m)$ and $\mathrm{Re}(\hat{\psi}_j) < 0$ $(j = 1, 2, \cdots, n)$, where $\tilde{\psi}_i$ and $\hat{\psi}_j$ are the eigenvalues of A and B, respectively. From Eq. (3.8), it is obvious that

$$\rho(E) = \max_{1 \leq i \leq m} \left| \frac{\psi + \tilde{\psi}_i}{\psi - \tilde{\psi}_i} \right| < 1, \quad \rho(F) = \max_{1 \leq j \leq n} \left| \frac{\psi + \hat{\psi}_j}{\psi - \hat{\psi}_j} \right| < 1.$$

By using the Kronecker products, from (3.11), we have

$$\mathrm{vec}(X_{k,j+1}) = \alpha F^T \otimes E \cdot \mathrm{vec}(X_{k,j}) + (1-\alpha) F^T \otimes E \cdot \mathrm{vec}(X_k) + \mathrm{vec}(Q). \tag{3.12}$$

Let $\mathrm{vec}(X_{k,j+1}) = x_{k,j+1},\ F^T \otimes E = H,\ \mathrm{vec}(X_k) = x_k$ and $\mathrm{vec}(Q) = \bar{q}$. From (3.12), it follows that

$$x_{k,j+1} = \alpha H x_{k,j} + (1-\alpha) H x_k + \bar{q}. \tag{3.13}$$

From (3.11) and (3.13), it is clear that

$$x_{k,j+1} = \left[(\alpha H)^{j+1} + (1-\alpha)\sum_{s=0}^{j}(\alpha H)^s H\right] x_k + \sum_{s=0}^{j}(\alpha H)^s \bar{q}.$$

Then

$$x_{k+1} = R_k x_k + G_k \bar{q}, \quad k = 0, 1, 2, \cdots, \tag{3.14}$$

where $R_k = (\alpha H)^{m_k} + (1-\alpha)\sum\limits_{s=0}^{m_k-1}(\alpha H)^s H$ and $G_k = \sum\limits_{s=0}^{m_k-1}(\alpha H)^s$.

Let $x^* = \mathrm{vec}(X^*)$. Since X^* is the exact solution to Eq. (3.8) , then

$$x^* = R_k x^* + G_k \bar{q}, \quad k = 0, 1, 2, \cdots. \tag{3.15}$$

Subtracting (3.15) from (3.14), then we obtain

$$x_{k+1} - x^* = R_k(x_k - x^*) = \cdots = R_k R_{k-1} \cdots R_0 (x_0 - x^*), \quad k = 0, 1, 2, \cdots, \tag{3.16}$$

and

$$\begin{aligned} R_k &= (\alpha H)^{m_k} + (1-\alpha)\sum_{s=0}^{m_k-1}(\alpha H)^s H \\ &= (\alpha H)^{m_k} + \sum_{s=0}^{m_k-1}(\alpha H)^s[(I-\alpha H) - (I-H)] \\ &= (\alpha H)^{m_k} + (I - (\alpha H)^{m_k}) - \sum_{s=0}^{m_k-1}(\alpha H)^s (I-H) \\ &= I - \sum_{s=0}^{m_k-1}(\alpha H)^s (I-H). \end{aligned} \tag{3.17}$$

Let λ_i be an eigenvalue of H. From (3.17), then

$$\phi_i^{(k)} = 1 - \frac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_k})}{1-\alpha\lambda_i} \tag{3.18}$$

is an eigenvalue of R_k.

Since $\rho(E) < 1$ and $\rho(F) < 1$, then we get

$$\rho(H) = \rho(E^T \otimes F) \le \rho(E)\rho(F) < 1,$$

thus $|\alpha\lambda_i| < 1$ for $0 < \alpha < 1$. From (3.18), it is clear that

$$\begin{aligned} \left|\phi_i^{(k)}\right| &= \left|1 - \tfrac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_k})}{1-\alpha\lambda_i}\right| \\ &= \left|\frac{\lambda_i\left(1-\alpha+\alpha^{m_k}\lambda_i^{m_k-1}-(\alpha\lambda_i)^{m_k}\right)}{1-\alpha\lambda_i}\right| \\ &< \tfrac{|1-\alpha|+|\alpha\lambda_i|^{m_k-1}|1-\lambda_i|}{|1-\alpha\lambda_i|} < 1 \end{aligned}$$

as $m_k \to \infty$. Then $\rho(R_k) < 1$.

Let $\chi = \max\limits_{k}\{\rho(R_k)\} < 1$ $(k = 0, 1, 2, \cdots)$ and $\hat{\phi}_i^{(k)}$ be an eigenvalue of $R_k R_{k-1} \cdots R_0$, then

$$\hat{\phi}_i^{(k)} = \Pi_{s=0}^{k} \left(1 - \frac{(1-\lambda_i)(1-(\alpha\lambda_i)^{m_s})}{1-\alpha\lambda_i}\right),$$

so

$$\rho(R_k R_{k-1} \cdots R_0) \leq \rho(R_k)\rho(R_{k-1}) \cdots \rho(R_0) \leq \chi^{k+1} < \bar{\gamma}^{k+1}$$

with $0 < \chi < \bar{\gamma} < 1$.

According to Lemma 6.5 [3], there exists an operator norm $\|\cdot\|_\zeta$ such that

$$\|R_k R_{k-1} \cdots R_0\|_\zeta < \bar{\gamma}^{k+1}.$$

From (3.16), we have

$$\|x_{k+1} - x^*\|_\zeta \leq \|R_k R_{k-1} \cdots R_0\|_\zeta \cdot \|x_0 - x^*\|_\zeta < \bar{\gamma}^{k+1}\|x_0 - x^*\|_\zeta. \tag{3.19}$$

Therefore, from (3.19), the iteration sequence $\{X_k\}_{k=0}^{\infty}$ generated by (3.11) converges to the exact solution X^* to Eq. (3.8) as $k \to \infty$.

From (3.18), it follows that

$$\begin{aligned}(1-\alpha\lambda_i)\phi_i^{(k)} &= (1-\alpha\lambda_i) - (1-\lambda_i)(1-(\alpha\lambda_i)^{m_k}) \\ &= \lambda_i(1-\alpha+\alpha^{m_k}\lambda_i^{m_k-1} - (\alpha\lambda_i)^{m_k}).\end{aligned} \tag{3.20}$$

From (3.20), then

$$\begin{aligned}\left|(1-\alpha\lambda_i)\phi_i^{(k)}\right| &= \left|\lambda_i(1-\alpha+\alpha^{m_k}\lambda_i^{m_k-1} - (\alpha\lambda_i)^{m_k})\right| \\ &= |\lambda_i| \cdot |1-\alpha+\alpha(\alpha\lambda_i)^{m_k-1}(1-\lambda_i)| \\ &< |\lambda_i|(|1-\alpha| + |\alpha\lambda_i|^{m_k-1} \cdot |1-\lambda_i|) \\ &= |\lambda_i| \cdot |1-\alpha| \\ &< |\lambda_i| \cdot |1-\alpha\lambda_i|\end{aligned} \tag{3.21}$$

with $m_k \to \infty$, then $|\phi_i^{(k)}| < |\lambda_i|$, so $\rho(R_k) < \rho(H)$ for $k = 0, 1, 2, \cdots$.

By using Smith method, the iteration sequence for solving Eq. (3.8) is

$$\tilde{X}_{k+1} = E\tilde{X}_k F + Q, \ k = 0, 1, 2, \cdots. \tag{3.22}$$

Let $\tilde{x}_{k+1} = \mathrm{vec}(\tilde{X}_{k+1})$, then from (3.22), we have

$$\tilde{x}_{k+1} = H\tilde{x}_k + \bar{q}, \ k = 0, 1, 2, \cdots. \tag{3.23}$$

Let $\{\tilde{x}_k\}_{k=0}^{\infty}$ be the iteration sequence derived from (3.23), and the initial vector $x^* - x_0$ be an eigenvector of H with eigenvalue λ_i and $|\lambda_i| = \rho(H)$. From (3.16), we obtain

$$\begin{aligned}
\|x_{k+1} - x^*\| &= \|R_k R_{k-1} \cdots R_0 (x_0 - x^*)\| \\
&= \|\phi_i^{(k)} \phi_i^{(k-1)} \cdots \phi_i^{(0)} (x_0 - x^*)\| \\
&\leq \|\rho(R_k)\rho(R_{k-1}) \cdots \rho(R_0)(x_0 - x^*)\| \ , \\
&< \|\rho(H)\rho(H) \cdots \rho(H)(x_0 - x^*)\| \\
&= \|H^{k+1}(x_0 - x^*)\| = \|\tilde{x}_{k+1} - x^*\|
\end{aligned}$$

then the IO iteration algorithm (3.11) converges faster than the Smith method with the same initial vector $x^* - x_0$.

In fact, for any initial vector, the IO iteration algorithm (3.11) always converges faster than the Smith method for solving Eq. (3.8). From Definition 6.5 [3], the convergence rate of $x_{m+1} = Rx_m + c$ is defined by

$$r(R) = -\log_{10}^{\rho(R)},$$

where $r(R)$ is the increase in the number of correct decimal places in the solution per iteration. Since $\rho(R_k) < \rho(H)$ $(k = 0, 1, 2, \cdots)$, it implies that the IO iteration algorithm has a higher convergence rate, i.e., the greater is the number of correct decimal places computed per iteration. Thus, the proof is completed. □

Remark 1. For the discrete Sylvester matrix equation [30]

$$X - AXB = C, \tag{3.24}$$

we can also solve it by using the IO iteration algorithm similar to Algorithm (3.11), and obtain the corresponding convergence theorem by referring to Theorem 3.1.

The algorithm for solving Eq. (3.24) can be described as follows:

$$\begin{cases}
X_{k,0} = X_k, X_0 = C, \ X_{k+1} = X_{k,\hat{m}_k}, \\
X_{k,j+1} = \hat{\alpha} A X_{k,j} B + (1 - \hat{\alpha}) A X_k B + C, \ k = 0, 1, 2, \cdots, \\
j = 0, 1, 2, \cdots, \hat{m}_k - 1
\end{cases} \tag{3.25}$$

with $0 < \hat{\alpha} < 1$.

4. The IO iteration algorithm for solving the CLMEs (2.4)

In this section, we will consider how to solve the CLMEs (2.4) by using the IO iteration algorithm proposed in Section 3. Just as (3.7), the CLMEs (2.4) have the following equivalent form:

$$(p_i I - \mathcal{A}_i^T) K_i (p_i I - \mathcal{A}_i) - (p_i I + \mathcal{A}_i^T) K_i (p_i I + \mathcal{A}_i) = 2p_i \left(\sum_{j=1, j\neq i}^{N} p_{ij} K_j + Q_i \right), \ i \in \Omega. \tag{4.1}$$

Since the Markovian jump linear system (2.1)-(2.2) is stochastically stable, from Lemma 2.2, it follows that the matrix $p_iI - \mathcal{A}_i$ is invertible for $p_i > 0$. Let $B_i = (p_iI - \mathcal{A}_i)^{-1}$, $V_i = (p_iI + \mathcal{A}_i)(p_iI - \mathcal{A}_i)^{-1}$ and $\bar{Q}_i = 2p_iB_i^T\left(\sum\limits_{j=1,j\neq i}^{N} p_{ij}K_j + Q_i\right)B_i$. From (4.1), then we have

$$K_i - V_i^T K_i V_i = \bar{Q}_i,\ i \in \Omega. \tag{4.2}$$

Now, we can apply the IO iteration algorithm (3.11) to solve each matrix equation in (4.2). The outer iteration sequences for solving (4.2) are given by

$$K_i(m+1) - \varphi_i V_i^T K_i(m+1)V_i = (1-\varphi_i)V_i^T K_i(m)V_i + \bar{Q}_i,\ m = 0, 1, 2, \cdots,\ i \in \Omega \tag{4.3}$$

with $0 < \varphi_i < 1$.

Let $W_i = (1-\varphi_i)V_i^T K_i(m)V_i + \bar{Q}_i$ and $Z_i = K_i(m+1)$. From (4.3), we obtain the following matrix equations

$$Z_i - \varphi_i V_i^T Z_i V_i = W_i,\ i \in \Omega. \tag{4.4}$$

The inner iteration sequences for solving (4.4) are

$$Z_i(j+1) = \varphi_i V_i^T Z_i(j)V_i + W_i,\ j = 0, 1, \cdots, \tilde{m}_k - 1,\ i \in \Omega \tag{4.5}$$

with the initial guess $Z_i(0) = K_i(m)$ and $Z_i(\tilde{m}_k)$ as the approximation of $K_i(m+1)$ in (4.3).

Let the relative residual

$$\varpi = \sum_{i=1}^{N} \frac{\left\| A_i^T K_i(m) + K_i(m)A_i + \sum\limits_{j=1}^{N} p_{ij}K_j(m) + Q_i \right\|_F}{\|Q_i\|_F}.$$

Algorithm 2: The IO iteration algorithm for solving the CLMEs (2.4)

Input: $V_i,\ \bar{Q}_i,\ \varphi_i,\ \gamma_i, \delta,\ i \in \Omega$

Output: K_i

1:**while** $\varpi \geq \delta$

2: **for** i=1:N

3: $K_i \leftarrow \bar{Q}_i$

4: $Z_i \leftarrow V_i^T K_i V_i$

5: **while** $\|\bar{Q}_i + Z_i - K_i\|_F \geq \gamma_i$

6: $W_i \leftarrow (1-\varphi_i)Z_i + \bar{Q}_i$

7: **for** s=1: $\tilde{m}_k$

8: $K_i \leftarrow \varphi_i Z_i + W_i$

9: $Z_i \leftarrow V_i^T K_i V_i$

10: **end**

11: **end**

12: **end**

13:**end**

4.1. A current-estimation-based IO iteration algorithm

From Algorithm 2, we find that the current estimate $K_i(m+1)$ for K_i is computed by only using the estimates $K_j(m)$ $(j \in \Omega,\ j \neq i)$ at the m-th step, while the estimates $K_j(m+1)$ $(j \in \Pi[1, i-1])$ have been obtained before $K_i(m+1)$ is updated. Thus, we can make use of both the estimates $K_1(m+1), \cdots, K_{i-1}(m+1)$ and $K_{i+1}(m), \cdots, K_N(m)$ to calculate the $K_i(m+1)$, just as the information renovation idea used in [8,11,19,23]. By the two-stage matrix splitting iteration framework similar to (3.11), we obtain the following current-estimation-based IO iteration algorithm:

$$\begin{cases} K_i(m,0) = K_i(m), K_i(0) = \hat{Q}_i, K_i(m+1) = K_i(m, \tilde{m}_k),\ i \in \Omega, \\ K_i(m, j+1) = \varphi_i V_i^T K_i(m,j) V_i + (1-\varphi_i) V_i^T K_i(m) V_i + \hat{Q}_i,\ m = 0, 1, 2, \cdots, \\ j = 0, 1, 2, \cdots, \tilde{m}_k - 1, \end{cases} \tag{4.6}$$

where

$$\hat{Q}_i = 2p_i B_i^T \left(\sum_{j=1}^{i-1} p_{ij} K_j(m+1) + \sum_{j=i+1}^{N} p_{ij} K_j(m) + Q_i \right) B_i.$$

Lemma 4.1. Assume that the Markovian jump linear system (2.1)-(2.2) is stochastically stable. If $p_i > 0$ and $0 < \varphi_i < 1$ for $i \in \Omega$, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ generated by (4.6) is upper bounded by the solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$ to the CLMEs (2.4) with zero initial conditions. Namely, for any integer $m \geq 0$, it follows that

$$K_i(m) < K_i,\ i \in \Omega. \tag{4.7}$$

Proof. Since the Markovian jump system (2.1)-(2.2) is stochastically stable, then the CLMEs (2.4) have unique positive definite solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$. Due to zero initial conditions, it is clear that $K_i(0) < K_i$ $(i \in \Omega)$. Now it is assumed that

$$K_i(l) < K_i,\ i \in \Omega \tag{4.8}$$

by the principle of the mathematical induction.

It follows from (4.6) that

$$\begin{aligned} K_i(l+1) \ &= \varphi_i^{\tilde{m}_k} (V_i^{\tilde{m}_k})^T K_i(l) V_i^{\tilde{m}_k} + (1-\varphi_i) \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (V_i^{s+1})^T K_i(l) V_i^{s+1} \\ &\quad + 2p_i \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \left(\sum_{j=1}^{i-1} p_{ij} K_j(l+1) + \sum_{j=i+1}^{N} p_{ij} K_j(l) + Q_i \right) B_i V_i^s,\ i \in \Omega \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} K_i \ &= \varphi_i^{\tilde{m}_k} (V_i^{\tilde{m}_k})^T K_i V_i^{\tilde{m}_k} + (1-\varphi_i) \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (V_i^{s+1})^T K_i V_i^{s+1} \\ &\quad + 2p_i \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \left(\sum_{j=1}^{i-1} p_{ij} K_j + \sum_{j=i+1}^{N} p_{ij} K_j + Q_i \right) B_i V_i^s,\ i \in \Omega. \end{aligned} \tag{4.10}$$

Subtracting (4.9) from (4.10), then

$$\begin{aligned}
&K_i - K_i(l+1)\\
&= \varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T(K_i - K_i(l))V_i^{\tilde{m}_k} + (1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(V_i^{s+1})^T(K_i - K_i(l))V_i^{s+1}\\
&+2p_i\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\left(\sum_{j=1}^{i-1}p_{ij}(K_j - K_j(l+1)) + \sum_{j=i+1}^{N}p_{ij}(K_j - K_j(l))\right)B_iV_i^s,\\
&i\in\Omega.
\end{aligned} \tag{4.11}$$

For $i=1$, $0<\varphi_1<1$ and $p_1>0$, from (4.11), then we have

$$\begin{aligned}
&K_1 - K_1(l+1)\\
&= \varphi_1^{\tilde{m}_k}(V_1^{\tilde{m}_k})^T(K_1 - K_1(l))V_1^{\tilde{m}_k} + (1-\varphi_1)\sum_{s=0}^{\tilde{m}_k-1}\varphi_1^s(V_1^{s+1})^T(K_1 - K_1(l))V_1^{s+1}\\
&+2p_1\sum_{s=0}^{\tilde{m}_k-1}\varphi_1^s(B_1V_1^s)^T\left(\sum_{j=2}^{N}p_{1j}(K_j - K_j(l))\right)B_1V_1^s > 0.
\end{aligned} \tag{4.12}$$

From (4.8) and (4.12), we obtain $K_1(l+1) < K_1$.

Now, we assume that

$$K_j(l+1) < K_j,\ j\in\Pi[1,t-1],\ t\geq 2. \tag{4.13}$$

For $i=t$, it follows from (4.11) that

$$\begin{aligned}
&K_t - K_t(l+1)\\
&= \varphi_t^{\tilde{m}_k}(V_t^{\tilde{m}_k})^T(K_t - K_t(l))V_t^{\tilde{m}_k} + (1-\varphi_t)\sum_{s=0}^{\tilde{m}_k-1}\varphi_t^s(V_t^{s+1})^T(K_t - K_t(l))V_t^{s+1}\\
&+2p_t\sum_{s=0}^{\tilde{m}_k-1}\varphi_t^s(B_tV_t^s)^T\left(\sum_{j=1}^{t-1}p_{tj}(K_j - K_j(l+1)) + \sum_{j=t+1}^{N}p_{tj}(K_j - K_j(l))\right)B_tV_t^s.
\end{aligned} \tag{4.14}$$

From (4.8), (4.13) and (4.14), it is obvious that $K_t(l+1) < K_t$ with $0<\varphi_t<1$ and $p_t>0$. By the induction principle, we have

$$K_i(l+1) < K_i,\ i\in\Omega. \tag{4.15}$$

From (4.8), (4.15) and the mathematical induction, we have $K_i(m) < K_i$ ($i\in\Omega$) for any integer $m\geq 0$. Thus, the proof is completed. □

Lemma 4.2. Assume that the Markovian jump linear system (2.1)-(2.2) is stochastically stable. If $p_i>0$ and $0<\varphi_i<1$ for $i\in\Omega$, then the matrix tuple $\mathcal{K}(m)=\{K_1(m),K_2(m),\cdots,K_N(m)\}$ derived from (4.6) with zero initial conditions is strictly monotonically increasing. Namely, for any integer $m\geq 0$, we have

$$K_i(m) < K_i(m+1),\ i\in\Omega. \tag{4.16}$$

Proof. From (4.9), it follows that

$$\begin{aligned} K_i(1) \quad &= \varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T K_i(0) V_i^{\tilde{m}_k} + (1-\varphi_i) \sum_{s=0}^{\tilde{m}_k-1} \varphi_i^s (V_i^{s+1})^T K_i(0) V_i^{s+1} \\ &+2p_i \sum_{s=0}^{\tilde{m}_k-1} \varphi_i^s (B_i V_i^s)^T \left(\sum_{j=1}^{i-1} p_{ij} K_j(1) + \sum_{j=i+1}^{N} p_{ij} K_j(0) + Q_i \right) B_i V_i^s, \ i \in \Omega. \end{aligned} \tag{4.17}$$

Since $K_i(0) = 0$ ($i \in \Omega$), then (4.17) can be rewritten as

$$K_i(1) = 2p_i \sum_{s=0}^{\tilde{m}_k-1} \varphi_i^s (B_i V_i^s)^T \left(\sum_{j=1}^{i-1} p_{ij} K_j(1) + Q_i \right) B_i V_i^s, \ i \in \Omega. \tag{4.18}$$

Since $p_1 > 0$, $Q_1 > 0$ and $0 < \varphi_1 < 1$, from (4.18), we have

$$K_1(1) = 2p_1 \sum_{s=0}^{\tilde{m}_k-1} \varphi_1^s (B_1 V_1^s)^T Q_1 B_1 V_1^s > 0,$$

then $K_1(0) < K_1(1)$.

For $i = 2$, from (4.18), it is clear that

$$\begin{aligned} K_2(1) \quad &= 2p_2 \sum_{s=0}^{\tilde{m}_k-1} \varphi_2^s (B_2 V_2^s)^T \left(p_{21} K_1(1) + Q_2 \right) B_2 V_2^s \\ &> 2p_2 \sum_{s=0}^{\tilde{m}_k-1} \varphi_2^s (B_2 V_2^s)^T \left(p_{21} K_1(0) + Q_2 \right) B_2 V_2^s \\ &= 2p_2 \sum_{s=0}^{\tilde{m}_k-1} \varphi_2^s (B_2 V_2^s)^T Q_2 B_2 V_2^s > 0 \end{aligned}$$

with $p_2 > 0$, $Q_2 > 0$ and $0 < \varphi_2 < 1$, then $K_2(0) < K_2(1)$. By repeating the above process, we can easily obtain that $K_i(0) < K_i(1), \ i \in \Pi[3, N]$.

Now, we assume that for $l \geq 1$,

$$K_i(l) < K_i(l+1), \ i \in \Omega. \tag{4.19}$$

It follows from (4.9) that

$$\begin{aligned} &K_i(l+2) - K_i(l+1) \\ &= \varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T (K_i(l+1) - K_i(l)) V_i^{\tilde{m}_k} + (1-\varphi_i) \sum_{s=0}^{\tilde{m}_k-1} \varphi_i^s (V_i^{s+1})^T (K_i(l+1) - K_i(l)) V_i^{s+1} \\ &+2p_i \sum_{s=0}^{\tilde{m}_k-1} \varphi_i^s (B_i V_i^s)^T \Bigg(\sum_{j=1}^{i-1} p_{ij} (K_j(l+2) - K_j(l+1)) + \sum_{j=i+1}^{N} p_{ij} (K_j(l+1) - K_j(l)) \Bigg) B_i V_i^s, \\ &i \in \Omega. \end{aligned} \tag{4.20}$$

Since $p_1 > 0$ and $0 < \varphi_1 < 1$, for $i = 1$, from (4.19) and (4.20), we have

$$\begin{aligned} &K_1(l+2) - K_1(l+1) \\ &= \varphi_1^{\tilde{m}_k}(V_1^{\tilde{m}_k})^T (K_1(l+1) - K_1(l)) V_1^{\tilde{m}_k} + (1-\varphi_1) \sum_{s=0}^{\tilde{m}_k-1} \varphi_1^s (V_1^{s+1})^T (K_1(l+1) - K_1(l)) V_1^{s+1} \\ &+2p_1 \sum_{s=0}^{\tilde{m}_k-1} \varphi_1^s (B_1 V_1^s)^T \Bigg(\sum_{j=2}^{N} p_{1j} \Big(K_j(l+1) - K_j(l) \Big) \Bigg) B_1 V_1^s > 0, \end{aligned}$$

it is clear that $K_1(l+1) < K_1(l+2)$.

Now, it is assumed that

$$K_j(l+1) < K_j(l+2), j \in \Pi[1, t-1]), \ t \geq 2. \tag{4.21}$$

According to (4.19) and (4.21), for $i = t$, we have

$$\begin{aligned}
&K_t(l+2) - K_t(l+1) \\
&= \varphi_t^{\tilde{m}_k}(V_t^{\tilde{m}_k})^T(K_t(l+1) - K_t(l))V_t^{\tilde{m}_k} + (1-\varphi_t)\sum_{s=0}^{\tilde{m}_k-1}\varphi_t^s(V_t^{s+1})^T(K_t(l+1) - K_t(l))V_t^{s+1} \\
&+2p_t\sum_{s=0}^{\tilde{m}_k-1}\varphi_t^s(B_tV_t^s)^T\left(\sum_{j=1}^{t-1}p_{tj}\Big(K_j(l+2) - K_j(l+1)\Big) + \sum_{j=t+1}^{N}p_{tj}\Big(K_j(l+1) - K_j(l)\Big)\right)B_tV_t^s \\
&> 0
\end{aligned} \tag{4.22}$$

with $p_t > 0$ and $0 < \varphi_t < 1$. From (4.22), we obtain that $K_t(l+1) < K_t(l+2)$. Then by the principle of mathematical induction, we have $K_i(l+1) < K_i(l+2)$ $(i \in \Omega)$. Therefore, for any integer $m \geq 0$, the relation (4.16) holds, and the proof is completed. □

Theorem 4.1. Assume that the Markovian jump linear system (2.1)-(2.2) is stochastically stable. If $p_i > 0, 0 < \varphi_i < 1$ for $i \in \Omega$, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ obtained from (4.6) converges to the unique solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$ to the CLMEs (2.4) with zero initial conditions. Namely, $\lim\limits_{m\to\infty} K_i(m) = K_i, \ i \in \Omega$.

Proof. From Lemmas 4.1 and 4.2, the iteration sequence $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ generated by (4.6) is monotonically increasing and upper bounded by the solutions to the CLMEs (2.4), then

$$0 = K_i(0) < K_i(1) < K_i(2) < \cdots < K_i(m) < K_i(m+1) < \cdots < K_i, \ i \in \Omega. \tag{4.23}$$

From [10], it is shown that the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ is convergent.

Let $\lim\limits_{m\to\infty} K_i(m) = K_i(\infty)$ $(i \in \Omega)$ and substitude $K_i(\infty)$ into (4.9), it is clear that

$$\begin{aligned}
K_i(\infty) \ &= \varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T K_i(\infty)V_i^{\tilde{m}_k} + (1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(V_i^{s+1})^T K_i(\infty)V_i^{s+1} \\
&+2p_i\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\left(\sum_{j=1}^{i-1}p_{ij}K_j(\infty) + \sum_{j=i+1}^{N}p_{ij}K_j(\infty) + Q_i\right)B_iV_i^s, \ i \in \Omega.
\end{aligned} \tag{4.24}$$

From (4.6), (4.24) are equivalent to

$$A_i^T K_i(\infty) + K_i(\infty)A_i + \sum_{j=1}^{N}p_{ij}K_j(\infty) + Q_i = 0, \ i \in \Omega. \tag{4.25}$$

From (4.25), it is obvious that $\mathcal{K}(\infty) = \{K_1(\infty), K_2(\infty), \cdots, K_N(\infty)\}$ is the solution to the CLMEs (2.4). Since the Markovian jump linear system (2.1)-(2.2) is stochastically stable, then the CLMEs (2.4) have a unique solution, so $\mathcal{K}(\infty) = \{K_1(\infty), K_2(\infty), \cdots, K_N(\infty)\}$ is the unique solution to the CLMEs (2.4) and $K_i(\infty) = K_i$ $(i \in \Omega)$. Thus, the proof is completed. □

From Theorem 1, it is stated that the proof is based on two assumptions: the stochastic stability of the associated Markovian jump linear system and zero initial conditions. Next, we give a convergence theorem for (4.6) without the assumption of zero initial conditions.

Theorem 4.2. Assume that the CLMEs (2.4) have a unique solution $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$. If $p_i > 0$ and $0 < \varphi_i < 1$ for $i \in \Omega$, then the matrix tuple $\mathcal{K}(m) = \{K_1(m), K_2(m), \cdots, K_N(m)\}$ generated by (4.6) converges to $\mathcal{K} = \{K_1, K_2, \cdots, K_N\}$ for any initial conditions if and only if $\mathcal{H}$ is invertible and $\rho(\mathcal{H}^{-1}\mathcal{F}) < 1$, where $\mathcal{H}$ is a lower triangular matrix with

$$\mathcal{H}_{ii} = I, \mathcal{H}_{ij} = -2p_i p_{ij} \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \otimes (B_i V_i^s)^T, j < i$$

and $\mathcal{F}$ is an upper triangular matrix with

$$\begin{cases} \mathcal{F}_{ii} = \varphi_i^{\tilde{m}_k} (V_i^{\tilde{m}_k})^T \otimes (V_i^{\tilde{m}_k})^T + (1 - \varphi_i) \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (V_i^{s+1})^T \otimes (V_i^{s+1})^T, \\ \mathcal{F}_{ij} = 2p_i p_{ij} \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \otimes (B_i V_i^s)^T, j > i. \end{cases}$$

Proof. From (4.9), we have

$$\begin{aligned} &\mathrm{vec}(K_i(m+1)) \\ &= \varphi_i^{\tilde{m}_k} (V_i^{\tilde{m}_k})^T \otimes (V_i^{\tilde{m}_k})^T \mathrm{vec}(K_i(m)) + (1 - \varphi_i) \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (V_i^{s+1})^T \otimes (V_i^{s+1})^T \mathrm{vec}(K_i(m)) \\ &+ 2p_i \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \otimes (B_i V_i^s)^T \left(\sum_{j=1}^{i-1} p_{ij} \mathrm{vec}(K_j(m+1)) + \sum_{j=i+1}^{N} p_{ij} \mathrm{vec}(K_j(m)) + \mathrm{vec}(Q_i) \right), \\ &i \in \Omega. \end{aligned} \tag{4.26}$$

Let

$$\theta(m) = \left(\mathrm{vec}(K_1(m))^T \ \ \mathrm{vec}(K_2(m))^T \cdots \mathrm{vec}(K_N(m))^T \right)^T,$$

$$\delta = \left(\mathrm{vec}(Q_1)^T \ \ \mathrm{vec}(Q_2)^T \cdots \mathrm{vec}(Q_N)^T \right)^T,$$

and Φ be a diagonal matrix with $\Phi(ii) = 2p_i \sum_{s=0}^{\tilde{m}_k - 1} \varphi_i^s (B_i V_i^s)^T \otimes (B_i V_i^s)^T$ $(i \in \Omega)$. Then (4.26) have the following equivalent form:

$$\mathcal{H}\theta(m+1) = \mathcal{F}\theta(m) + \Phi\delta. \tag{4.27}$$

Since $\mathcal{H}$ is an invertible matrix, it follows from (4.27) that

$$\theta(m+1) = \mathcal{H}^{-1}\mathcal{F}\theta(m) + \mathcal{H}^{-1}\Phi\delta. \tag{4.28}$$

From (4.28), we obtain the following recursive relation:

$$\theta(m+1) = (\mathcal{H}^{-1}\mathcal{F})^{m+1}\theta(0) + \sum_{i=0}^{m} (\mathcal{H}^{-1}\mathcal{F})^i \mathcal{H}^{-1}\Phi\delta. \tag{4.29}$$

Since $\rho(\mathcal{H}^{-1}\mathcal{F}) < 1$, then

$$\begin{aligned}\lim_{m\to\infty}\theta(m+1) &= \lim_{m\to\infty}\left((\mathcal{H}^{-1}\mathcal{F})^{m+1}\theta(0) + \sum_{i=0}^{m}(\mathcal{H}^{-1}\mathcal{F})^{i}\mathcal{H}^{-1}\Phi\delta\right)\\ &= (I-\mathcal{H}^{-1}\mathcal{F})^{-1}\mathcal{H}^{-1}\Phi\delta = (\mathcal{H}-\mathcal{F})^{-1}\Phi\delta.\end{aligned}$$

Let $\theta = \left(\mathrm{vec}(K_1)^T \;\; \mathrm{vec}(K_2)^T \cdots \mathrm{vec}(K_N)^T\right)^T$. From (4.10), we get

$$\mathcal{H}\theta = \mathcal{F}\theta + \Phi\delta. \tag{4.30}$$

According to (4.30), the exact solution to the CLMEs (2.4) is

$$\theta = (\mathcal{H}-\mathcal{F})^{-1}\Phi\delta.$$

Thus,

$$\lim_{m\to\infty}\theta(m+1) = \theta,$$

and the proof is completed. □

4.2. Two weighted IO iteration algorithms

In this section, we propose two weighted IO iteration algorithms for solving the CLMEs (2.4). Analogous to the accelerated Smith iteration algorithm [11], we rewrite (4.2) in the following equivalent form:

$$K_i - V_i^T K_i V_i = 2B_i^T\left(\sum_{j=1}^{i-1} p_{ij}\Big((p_i-1)K_j + K_j\Big) + \sum_{j=i+1}^{N} p_i p_{ij} K_j + p_i Q_i\right) B_i,\ i\in\Omega. \tag{4.31}$$

If we solve (4.31) by Algorithm (4.6), then obtain the first weighted IO iteration algorithm as follows:

$$\begin{aligned}&K_i(m+1)\\ &= \varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T K_i(m) V_i^{\tilde{m}_k} + (1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s (V_i^{s+1})^T K_i(m) V_i^{s+1}\\ &+2\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s (B_i V_i^s)^T\left(\sum_{j=1}^{i-1} p_{ij}\Big((p_i-1)K_j(m+1) + K_j(m)\Big) + \sum_{j=i+1}^{N} p_i p_{ij} K_j(m) + p_i Q_i\right) B_i V_i^s,\\ &i\in\Omega.\end{aligned} \tag{4.32}$$

According to Algorithm (26) [19], we reformulate (4.2) in the following way:

$$K_i - V_i^T K_i V_i = 2p_i B_i^T\left(\sum_{j=1}^{i-1} p_{ij}\Big((1-\omega)K_j + \omega K_j\Big) + \sum_{j=i+1}^{N} p_{ij} K_j + Q_i\right) B_i,\ i\in\Omega \tag{4.33}$$

with $0 \le \omega < 1$.

Using Algorithm (4.6) to solve (4.33), then we derive the second weighted IO iteration algorithm:

$$
\begin{aligned}
&K_i(m+1)\\
&=\varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T K_i(m) V_i^{\tilde{m}_k}+(1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(V_i^{s+1})^T K_i(m)V_i^{s+1}\\
&+2p_i\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\left(\sum_{j=1}^{i-1}p_{ij}\Big((1-\omega)K_j(m+1)+\omega K_j(m)\Big)+\sum_{j=i+1}^{N}p_{ij}K_j(m)+Q_i\right)B_iV_i^s,\\
&i\in\Omega,
\end{aligned}
\tag{4.34}
$$

Theorem 4.3. If the CLMEs (2.4) have a unique solution $\mathcal{K}=\{K_1,K_2,\cdots,K_N\}$, $p_i>0$ and $0<\varphi_i<1$ $(i\in\Omega)$, then the matrix tuple $\mathcal{K}(m)=\{K_1(m),K_2(m),\cdots,K_N(m)\}$ derived from (4.32) converges to $\mathcal{K}=\{K_1,K_2,\cdots,K_N\}$ for any initial conditions if and only if $\tilde{\mathcal{H}}$ is invertible and $\rho(\tilde{\mathcal{H}}^{-1}\tilde{\mathcal{F}})<1$, where $\tilde{\mathcal{H}}$ is a lower triangular matrix with

$$
\tilde{\mathcal{H}}_{ii}=I,\tilde{\mathcal{H}}_{ij}=-2(p_i-1)p_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j<i
$$

and $\tilde{\mathcal{F}}$ is a matrix with

$$
\begin{cases}
\tilde{\mathcal{F}}_{ii}=\varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T\otimes(V_i^{\tilde{m}_k})^T+(1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(V_i^{s+1})^T\otimes(V_i^{s+1})^T,\\
\tilde{\mathcal{F}}_{ij}=2p_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j<i,\\
\tilde{\mathcal{F}}_{ij}=2p_ip_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j>i.
\end{cases}
$$

Proof. The proof is similar to that of Theorem 4.2. □

Theorem 4.4. Assume that the CLMEs (2.4) have a unique solution $\mathcal{K}=\{K_1,K_2,\cdots,K_N\}$, If $p_i>0$ and $0<\varphi_i<1$ for $i\in\Omega$, then the matrix tuple $\mathcal{K}(m)=\{K_1(m),K_2(m),\cdots,K_N(m)\}$ generated by (4.34) converges to $\mathcal{K}=\{K_1,K_2,\cdots,K_N\}$ for any initial conditions if and only if $\hat{\mathcal{H}}$ is invertible and $\rho(\hat{\mathcal{H}}^{-1}\hat{\mathcal{F}})<1$, where $\hat{\mathcal{H}}$ is a lower triangular matrix with

$$
\hat{\mathcal{H}}_{ii}=I,\hat{\mathcal{H}}_{ij}=-2(1-\omega)p_ip_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j<i
$$

and $\hat{\mathcal{F}}$ is a matrix with

$$
\begin{cases}
\hat{\mathcal{F}}_{ii}=\varphi_i^{\tilde{m}_k}(V_i^{\tilde{m}_k})^T\otimes(V_i^{\tilde{m}_k})^T+(1-\varphi_i)\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(V_i^{s+1})^T\otimes(V_i^{s+1})^T,\\
\hat{\mathcal{F}}_{ij}=2\omega p_ip_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j<i,\\
\hat{\mathcal{F}}_{ij}=2p_ip_{ij}\sum_{s=0}^{\tilde{m}_k-1}\varphi_i^s(B_iV_i^s)^T\otimes(B_iV_i^s)^T,j>i.
\end{cases}
$$

Proof. The proof can be completed by a similar way to Theorem 4.2. □

Compared with the previous algorithms (2.5)-(2.9), we have some remarks given as follows:

Remark 2. For the proposed IO iteration algorithms in this paper, we only need to compute $(p_iI - \mathcal{A}_i)^{-1}$ $(i \in \Omega)$ in the first iteration step, which computational cost is much less than the overall cost of the algorithms (4.6), (4.32) and (4.34), respectively.

Remark 3. In regard to the implicit algorithms (2.5) and (2.6), by using our proposed explicit iteration algorithms, the iteration solutions to the CLMEs (2.4) can be obtained explicitly in each iteration step. Therefore, the algorithms (4.6), (4.32) and (4.34) should take less CPU time than the algorithms (2.5) and (2.6), especially for large continuous-time Markovian jump linear systems, which is illustrated by the numerical results in Section 6.

Remark 4. In comparison our proposed iteration algorithms with the explicit algorithms (2.7)-(2.9), the algorithms (4.6), (4.32), (4.34) perform better than the algorithms (2.7)-(2.9) in terms of the iteration steps and CPU time according to the simulation results in Section 6.

5. The choices of the parameters in the IO iteration algorithms

In this section, we will discuss the choices of the parameters in the proposed iteration algorithms. Since it is difficult to obtain the optimal parameters, then we can only give some heuristical strategies for the choices of the corresponding parameters in the following sections.

5.1. The choices of the parameters in Algorithms (3.11) and (3.25)

In this section, we consider the choices of the parameters α, m_k and ψ in the IO iteration algorithm (3.11). By the Kronecker products, the outer iteration sequence (3.9) can be reformulated as

$$(I - \alpha F^T \otimes E)x_{k+1} = (1-\alpha)(F^T \otimes E)x_k + \bar{q}, \tag{5.1}$$

and the inner iteration sequence (3.10) has the following form:

$$y_{j+1} = (\alpha F^T \otimes E)y_j + w, \tag{5.2}$$

where $y_{j+1} = \mathrm{vec}(Y_{j+1})$ and $w = \mathrm{vec}(W)$, respectively.

It is clear that the outer iteration sequence (5.1) is based on the following matrix splitting

$$I - F^T \otimes E = M_1 - N_1,\ M_1 = I - \alpha F^T \otimes E,\ N_1 = (1-\alpha)F^T \otimes E,$$

and the corresponding iteration matrix

$$\bar{R} = M_1^{-1}N_1 = (1-\alpha)(I - \alpha F^T \otimes E)^{-1}(F^T \otimes E). \tag{5.3}$$

The inner iteration sequence (5.2) is associated with the matrix splitting

$$M_1 = M_2 - N_2,\ M_2 = I,\ N_2 = \alpha F^T \otimes E,$$

and the corresponding iteration matrix is

$$\hat{R} = M_2^{-1} N_2 = \alpha F^T \otimes E. \tag{5.4}$$

Let λ_i be an eigenvalue of $F^T \otimes E$. Assume $0 < \alpha < 1$ and $\rho(E)\rho(F) < 1$, then $|\lambda_i| < 1$. From (5.3) and (5.4), we have

$$\rho(\bar{R}) = \max_i \left| \frac{(1-\alpha)\lambda_i}{1-\alpha\lambda_i} \right| \leq \max_i \frac{(1-\alpha)|\lambda_i|}{1-\alpha|\lambda_i|} = \frac{(1-\alpha)\rho(F^T \otimes E)}{1-\alpha\rho(F^T \otimes E)} \tag{5.5}$$

and

$$\rho(\hat{R}) = \max_i |\alpha\lambda_i| = \alpha\rho(F^T \otimes E). \tag{5.6}$$

Let $g(\alpha) = \frac{(1-\alpha)\rho(F^T \otimes E)}{1-\alpha\rho(F^T \otimes E)}$, by simple calculation, then we obtain

$$g'(\alpha) = \frac{\rho(F^T \otimes E)(\rho(F^T \otimes E) - 1)}{(1-\alpha\rho(F^T \otimes E))^2} < 0.$$

Hence, $g(\alpha)$ is monotonically decreasing, so the outer iteration sequence (3.9) converges faster for a larger α. From (5.6), it is clear that the inner iteration sequence (3.10) converges faster if α is close to zero. Thus, how to determine α for accelerating the IO iteration algorithm and reducing its computational work as much as possible is a challenge. From (5.5), we find that an appropriate parameter α only reduces the upper bound of the spectral radius $\rho(\bar{R})$, but does not decrease the spectral radius itself. For many numerical examples, the values of parameter α around 0.5 can achieve better numerical results, which is verified in Section 6.

For the parameter m_k, just as the analyses in [4], if the value of m_k is large, which may spend a long computational time performing inner iterations, just to compute a single outer iteration at a time, and slow the overall convergence rate. On the other hand, if m_k is small, which may result in inner iterations that do not sufficiently approximate the exact solution to the matrix equation $Y - \alpha EYF = W$, and hence do not yield sufficient progress towards the exact solution to Eq. (3.8). In Section 6, we observe that the better numerical results can be achieved with a smaller m_k, such as $m_k = 2, 3$.

For the parameter ψ in Eq. (3.8), we learn from (5.5) and (5.6) that smaller the $\rho(E)\rho(F)$, faster the convergence of the IO iteration algorithm is. The optimal ψ satisfies the following relation:

$$\min_{\psi>0} \max_{\tilde{\psi}_i, \hat{\psi}_j} \frac{|\psi + \tilde{\psi}_i|}{|\psi - \tilde{\psi}_i|} \cdot \frac{|\psi + \hat{\psi}_j|}{|\psi - \hat{\psi}_j|}, \tag{5.7}$$

where $\tilde{\psi}_i$ $(i = 1, 2, \cdots, m), \hat{\psi}_j$ $(j = 1, 2, \cdots, n)$ are eigenvalues of A and B, respectively. For the general matrices A and B, it is not easy to find the optimal ψ. However, for some special cases, the optimal ψ can be obtained from (5.7). For example, if Eq. (3.8) is the Lyapunov matrix equation

$$A^T X + XA = Q, \tag{5.8}$$

and all the eigenvalues of A are negative real numbers, then the following result holds.

Theorem 5.1 [31]. Assume that all the eigenvalues of A in Eq. (5.8) are negative real numbers and $\psi > 0$. Let $\tilde{\psi}_{\min}$ and $\tilde{\psi}_{\max}$ be the minimum and maximum eigenvalues of A, respectively. Then the optimal ψ is

$$\bar{\psi} = \arg\min_{\psi} \left\{ \max_{\tilde{\psi}_k} \frac{|\psi + \tilde{\psi}_k|}{|\psi - \tilde{\psi}_k|} \right\} = \sqrt{\tilde{\psi}_{\min}\tilde{\psi}_{\max}}.$$

For the choices of the parameters $\hat{\alpha}$ and $\hat{m}_k$ in Algorithm (3.25), the similar conclusions can be drawn according to the analyses of the parameters α and m_k, respectively.

5.2. The choices of the parameters in Algorithms 2, (4.6), (4.32) and (4.34)

From the analyses in Section 4, when using Algorithms 2, (4.6), (4.32) and (4.34) to solve the CLMEs (2.4), we need to solve N Lyapunov matrix equations in each iteration step. Therefore, for the choices of the parameters p_i, φ_i and $\tilde{m}_k$ in each Lyapunov matrix equation, we can get the similar conclusions to the parameters ψ, α and m_k, respectively. However, for Algorithm (4.32), the p_i $(i \in \Omega)$ calculated by Theorem 5.1 may not be optimal, since p_i $(i \in \Omega)$ are also treated as the relax factors in the right-hand side of the iteration sequences (4.32), thus we can only obtain the appropriate p_i $(i \in \Omega)$ through numerical experiments.

6. Numerical results

In this section, we present several numerical examples to illustrate the performances of the proposed algorithms for solving Eqs. (3.6), (3.24) and the CLMEs (2.4), respectively. Three iteration parameters are used to test these algorithms, which are iteration step (denoted as IT), computing time in seconds (denoted as CPU), and relative residual (denoted as RES), where the RES for the CLMEs (2.4) is

$$\sum_{i=1}^{N} \frac{\left\| A_i^T K_i(m) + K_i(m) A_i + \sum_{j=1}^{N} p_{ij} K_j(m) + Q_i \right\|_F}{\|Q_i\|_F},$$

, the RES for Eq. (3.6) is $\frac{\|\tilde{R}^{(k)}\|_F}{\|C\|_F}$ with $\tilde{R}^{(k)} = AX_k + X_k B - C$, the RES for Eq. (3.24) is $\frac{\|\vec{R}^{(k)}\|_F}{\|C\|_F}$ with $\vec{R}^{(k)} = X_k - AX_k B - C$, where k and m denote the iteration number, respectively.

Example 1. Consider the matrix equation $AX + XB = C$, where $A(i,i) = -2.5,\ A(i,i+1) = 1,\ A(i+1,i) = -3,\ A(i+2,i) = -3,\ A(i,i+2) = 1.\ B = A^T$ and $C_{i,j} = 1$ for $i, j = 1, 2, \cdots, n$.

In this example, we compare Algorithm (3.11) with the Smith method [30], where $\alpha = 0.7$, $\psi = 4$ and $m_k = 2$, respectively.

From Fig. 1 and Table 1, it is clear that Algorithm (3.11) converges faster than the Smith method in terms of iteration step and CPU time, especially for large matrix equations, such as $n = 800$.

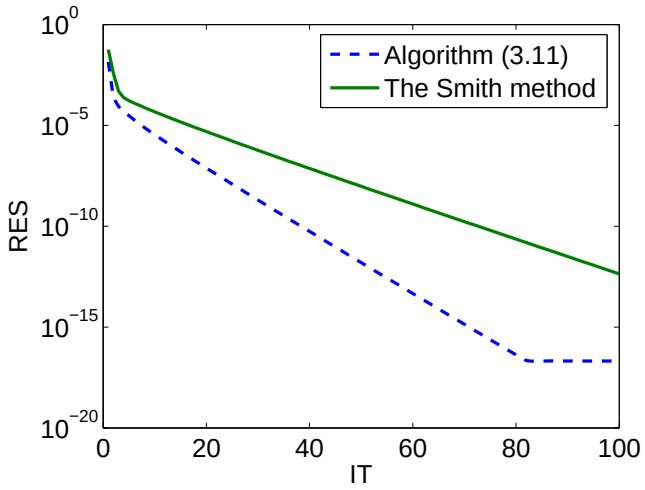

Figure 1: Convergence curves with IT=100 and n=600

Table 1: Numerical results of Algorithm (3.11) and Smith method

	The Smith method			Algorithm (3.11)		
n	IT	CPU	RES	IT	CPU	RES
50	33	0.01095	1.30×10^{-9}	18	0.00519	1.83×10^{-9}
300	59	1.12446	3.09×10^{-9}	31	0.66575	2.78×10^{-9}
500	59	4.31727	1.85×10^{-9}	31	2.27159	1.67×10^{-9}
800	60	14.4988	1.16×10^{-9}	33	7.93633	1.04×10^{-9}

Example 2. In this example, consider the solution of the following discrete Lyapunov matrix equation $X - AXA^T = C$, where A is defined as in [13] and

$$A = \begin{bmatrix} 0 & \nu & & & & \\ -\nu & 0 & \nu & & & \\ & -\nu & 0 & \cdot & & \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot & \nu \\ & & & & & -\nu & 0 \end{bmatrix},$$

$C_{i,j} = 1$ for $i, j = 1, 2, \cdots, n$. The eigenvalues of A are given by $\lambda_j = 2i|\nu| \cos \frac{\pi j}{n+1}$, $j = 1, \cdots, n$. When n is large and ν is close to 0.5, $\rho(A)$ becomes close to 1, which slows down the convergence rate of the Smith method greatly.

In this example, we make a comparison of Algorithm (3.25) with the Smith method, where $n = 500$, $\hat{\alpha} = 0.6$ and $\hat{m}_k = 2$, respectively. From Table 2, it is clear that the Smith method needs more iteration steps and CPU time than Algorithm (3.25) in both the iteration steps and CPU time. For the larger ν, Algorithm (3.25) is more efficient than the Smith method, such as the cases for $\nu = 0.49$ and 0.495, respectively.

Table 2: Numerical results of Algorithm (3.25) and Smith method

	The Smith method			Algorithm (3.25)		
ν	IT	CPU	RES	IT	CPU	RES
0.40	19	1.44211	1.44×10^{-9}	10	0.95560	9.83×10^{-9}
0.42	23	1.62581	1.70×10^{-9}	14	1.13830	1.48×10^{-9}
0.44	30	2.04436	1.62×10^{-9}	18	1.39988	1.45×10^{-9}
0.46	42	3.03121	1.89×10^{-9}	25	1.92395	1.62×10^{-9}
0.49	130	9.30354	1.99×10^{-9}	74	5.51254	1.88×10^{-9}
0.495	222	16.0229	1.97×10^{-9}	123	9.28304	1.98×10^{-9}

Now, we investigate the choice of the parameter $\hat{m}_k$ in Algorithm (3.25) with $n = 500$, $\nu = 0.48$ and $\hat{\alpha} = 0.7$. From Table 3, it follows that Algorithm (3.25) with a larger $\hat{m}_k$ may need less iteration steps, but take more computing time to satisfy the given stopping criterion, for example, the cases for $\hat{m}_k = 6, 7$ compared with those for $\hat{m}_k = 3, 4$. Then Algorithm (3.25) with a smaller $\hat{m}_k$ can achieve better numerical results, which is consistent with the analyses in Section 5.

Next, we perform Algorithm (3.25) with different $\hat{\alpha}$, where $\hat{m}_k = 3$, $\hat{\alpha} = \frac{i}{10}$, $i = 2, \cdots, 8$. From Table 4, we find that Algorithm (3.25) has worse effectiveness for a larger $\hat{\alpha}$ or smaller $\hat{\alpha}$, such as the case $\hat{\alpha} = 0.8$ or 0.2, and Algorithm (3.25) performs better when $\hat{\alpha}$ is around 0.5, which is also in concord with our conclusions in Section 5.

Table 3: Numerical results of Algorithm (3.25) for different $\hat{m}_k$.

$\hat{m}_k$	IT	CPU	RES
2	41	6.53207	1.95×10^{-9}
3	32	6.58435	1.80×10^{-9}
4	28	6.67555	1.83×10^{-9}
5	26	7.19918	1.80×10^{-9}
6	25	7.47139	1.71×10^{-9}
7	24	7.92004	1.87×10^{-9}

Table 4: Numerical results of Algorithm (3.25) for different $\hat{\alpha}$.

$\hat{\alpha}$	IT	CPU	RES
0.2	56	8.86911	1.82×10^{-9}
0.3	50	8.37723	1.87×10^{-9}
0.4	45	7.68162	1.78×10^{-9}
0.5	40	7.04392	1.84×10^{-9}
0.6	36	6.77274	1.74×10^{-9}
0.7	42	7.35351	1.80×10^{-9}
0.8	49	7.78312	1.63×10^{-9}

Example 3. In this example, consider the CLMEs (2.3) with system matrices and transition rate matrix defined as follows:

$$A_1 = \begin{bmatrix} -1.3232 & -1.1582 & 1.0290 \\ -0.12292 & -2.0737 & 0.2234 \\ -0.6075 & 1.1656 & -3.1031 \end{bmatrix}, \ A_2 = \begin{bmatrix} -2.479 & 1.3537 & -0.5717 \\ 0.8246 & -1.8727 & 0.4868 \\ 1.0958 & -0.9525 & -0.6483 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -2.7604 & 0.5164 & -0.0381 \\ 0.5067 & -2.6064 & 0.399 \\ 0.528 & -0.2465 & -2.1332 \end{bmatrix}, \ P = \begin{bmatrix} -4 & 3 & 1 \\ 2 & -2.5 & 0.5 \\ 1.75 & 1.75 & -3.5 \end{bmatrix}.$$

The system has been investigated in [11,12], and the number of the subsystems is $N = 3$. We choose the positive definite matrices Q_i $(i \in \Pi[1,3])$ as identity matrices.

First, we make a comparison for Algorithms 2, (4.6), (4.32), (4.34) and the iteration algorithms in [11]. Let the parameters $p_1,\ p_2,\ p_3$ in Algorithm 2, (4.6), (4.32), (4.34) and the parameter q in [11] satisfy the relation $p_1 = p_2 = p_3 = |q|$. We choose $\tilde{m}_k = 2$ and $\varphi_i = 0.7$ $(i \in \Pi[1,3])$, and list the numerical results in Tables 5, 6, 7 for different $|q|$, respectively. From Table 5, it follows that Algorithm 2 outperforms the Smith method for solving the CLMEs (2.4) in terms of the iteration steps and CPU time. Moreover, Algorithm 2 has more effectiveness than the Smith method for a larger $|q|$. In Table 6, we compare Algorithm (4.6) with Algorithm (2.8), the similar conclusions can be drawn from Table 6 as those from Table 5. In Table 7, we compare Algorithms (4.32), (4.34) with Algorithm (2.9) for $\omega = 0.2$, respectively. The two weighted IO iteration algorithms (4.32), (4.34) both converge faster than Algorithm (2.9) in both iteration steps and CPU time. Furthermore, Algorithm (4.32) performs best for the larger $|q|$.

Table 5: Numerical results of Algorithm 2 and the Smith method

	The Smith method			Algorithm 2		
$\|q\|$	IT	CPU	RES	IT	CPU	RES
1	81	0.006590	8.23×10^{-15}	64	0.003866	7.34×10^{-15}
5	67	0.004448	8.40×10^{-15}	53	0.004371	8.96×10^{-15}
10	82	0.003034	8.03×10^{-15}	64	0.002988	9.57×10^{-15}
15	110	0.003615	8.38×10^{-15}	79	0.001958	7.82×10^{-15}
20	139	0.003927	9.37×10^{-15}	95	0.003284	8.19×10^{-15}
25	169	0.003183	9.28×10^{-15}	112	0.002741	8.41×10^{-15}

Since the eigenvalues of A_i $(i \in \Pi[1,3])$ are all negative real numbers, then we can obtain the optimal p_i denoted as $\hat{p}_i$ $(i \in \Pi[1,3])$ by Theorem 5.1, where $\varphi_i = 0.7$ $(i \in \Pi[1,3])$, $\tilde{m}_k = 2$ and $\omega = 0.1$ in Algorithms (4.6) and (4.34). From Figs. 2, 3 and Table 8, it is clear that the algorithms with the optimal parameters $\hat{p}_i$ $(i \in \Pi[1,3])$ perform better compared with other values of the parameters p_i $(i \in \Pi[1,3])$. Although Algorithm (4.6) has the same iteration steps for $|q| = 4$ and the optimal parameters $\hat{p}_i$ $(i \in \Pi[1,3])$, it needs more CPU time for $|q| = 4$ at the same time. Furthermore, Algorithm (4.6) converges faster than Algorithm (4.34) with the same optimal parameters $\hat{p}_i$ $(i \in \Pi[1,3])$.

Table 6: Numerical results of Algorithms (2.8) and (4.6)

	Algorithm (2.8)			Algorithm (4.6)		
$\|q\|$	IT	CPU	RES	IT	CPU	RES
1	64	0.001230	8.31×10^{-15}	43	0.001091	5.33×10^{-15}
5	44	0.002717	7.71×10^{-15}	29	0.002108	8.51×10^{-15}
10	62	0.001229	7.00×10^{-15}	43	0.001107	5.09×10^{-15}
15	91	0.001912	8.91×10^{-15}	59	0.001662	7.34×10^{-15}
20	121	0.003693	8.65×10^{-15}	75	0.002971	9.95×10^{-15}
25	151	0.002866	8.84×10^{-15}	93	0.002465	7.66×10^{-15}

Table 7: Numerical results of Algorithms (2.9), (4.32) and (4.34)

	Algorithm (2.9)			Algorithm (4.32)			Algorithm (4.34)		
$\|q\|$	IT	CPU	RES	IT	CPU	RES	IT	CPU	RES
2	47	0.001055	6.14×10^{-15}	43	0.001228	6.60×10^{-15}	36	0.001030	5.12×10^{-15}
7	48	0.001716	8.29×10^{-15}	38	0.000992	8.01×10^{-15}	40	0.001050	5.68×10^{-15}
12	75	0.001499	8.85×10^{-15}	51	0.001418	7.96×10^{-15}	54	0.001408	7.66×10^{-15}
17	105	0.002131	7.37×10^{-15}	67	0.001735	7.99×10^{-15}	70	0.001829	7.58×10^{-15}
22	134	0.002970	8.79×10^{-15}	84	0.002426	7.70×10^{-15}	87	0.002445	8.03×10^{-15}
25	164	0.003507	9.17×10^{-15}	100	0.002603	9.86×10^{-15}	104	0.002750	9.01×10^{-15}

Table 8: Numerical results of Algorithms (4.6), (4.34) for different parameters

	Algorithm (4.6)			Algorithm (4.34)		
$\|q\|$	IT	CPU	RES	IT	CPU	RES
4	28	0.001294	5.28×10^{-15}	35	0.000953	5.29×10^{-15}
9	40	0.001030	5.54×10^{-15}	45	0.001187	7.39×10^{-15}
14	55	0.001418	9.10×10^{-15}	60	0.001637	6.88×10^{-15}
19	72	0.001719	9.27×10^{-15}	76	0.001977	9.91×10^{-15}
$\hat{p_1}, \hat{p_2}, \hat{p_3}$	28	0.000684	8.09×10^{-15}	32	0.000839	7.98×10^{-15}

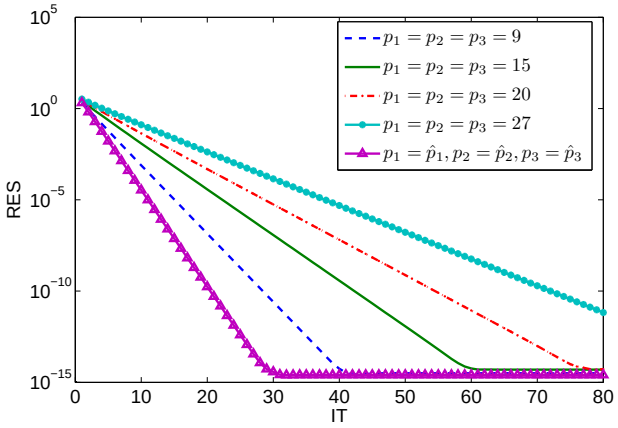

Figure 2: Convergence curves of Algorithm (4.6) with different parameters

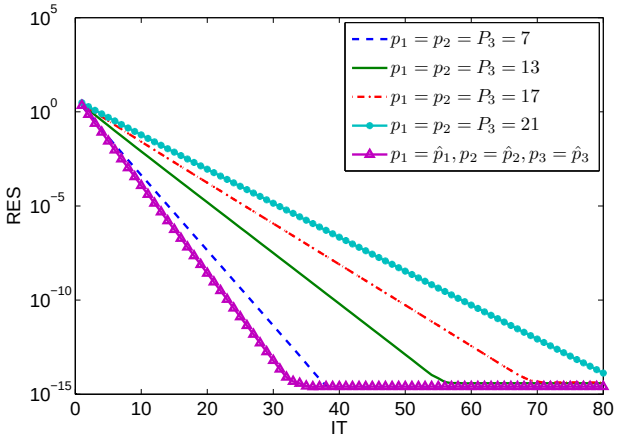

Figure 3: Convergence curves of Algorithm (4.34) with different parameters

Next, we will compare the convergence performances of Algorithms (4.6), (4.32), (4.34) with Algorithms 2.5, 2.6, 2.7, respectively. All algorithms are started with the following initial conditions:

$$K_1(0) = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 0 & 1.2 \\ 2 & -3 & 0.8 \end{bmatrix}, K_2(0) = \begin{bmatrix} -1 & 0.5 & 0.7 \\ 1 & 0 & 0.9 \\ 0 & 2.1 & -1 \end{bmatrix}, K_3(0) = \begin{bmatrix} 0.8 & -0.5 & 1.6 \\ 0.15 & 2.3 & -0.7 \\ 0.3 & -2.1 & 1.5 \end{bmatrix}.$$

Here, we choose $\varphi_i = 0.8, p_i = 4$ $(i \in \Pi[1,3])$, $\tilde{m}_k = 2$ and $\omega = 0.1$, respectively. Let $\mu = 0.0114$ in Algorithm (2.7). From Table 9, we find that Algorithm (2.5) needs fewest iteration steps, however, it also takes more CPU time than Algorithms (4.6), (4.32), (4.34), respectively. Moreover, Algorithms (2.5), (2.6) are implicit algorithms, so it is difficult to solve N Lyapunov matrix equations in each iteration step for large CLMEs (2.4). Due to this, the explicit algorithms (4.6), (4.32), (4.34) proposed in this paper are more efficient for large CLMEs (2.4).

Table 9: Comparison of the convergence results for different algorithms.

Iteration algorithm	IT	CPU	RES
Algorithm (2.5)	25	0.028055	4.12×10^{-14}
Algorithm (2.6)	31	0.012804	8.01×10^{-14}
Algorithm (2.7)	403	0.009850	9.54×10^{-14}
Algorithm (4.6)	26	0.000670	4.54×10^{-14}
Algorithm (4.32)	33	0.000914	4.00×10^{-14}
Algorithm (4.34)	29	0.000855	7.64×10^{-14}

7.Conclusions

In this paper, an IO iteration algorithm for solving some matrix equations is presented. The algorithm is firstly employed to solve the Sylvester matrix equation, and the corresponding convergence is analyzed. Next, the IO iteration algorithm is used to solve the CLMEs (2.4). In order to improve the convergence rate of the IO iteration algorithm, a current-estimation-based IO iteration algorithm is constructed by introducing the latest estimation. Afterwards, two weighted IO iteration algorithms are proposed, and the corresponding convergence theorems are given. Finally, several numerical examples demonstrate the efficiency of the proposed algorithms. Since these algorithms are rather parameter-dependent, thus how to determine the optimal parameters is an interesting topic, and will be further investigated in our future work.

Acknowledgements

The authors sincerely thank the anonymous referees for their constructive and valuable comments, which greatly improved the presentation of this paper. The first author acknowledges the partial support of the China Scholarship Council (Grant No. 201706935029).

References

[1] X. Feng , K.A. Loparo , Y. Ji , H.J. Chizeck , Stochastic stability properties of jump linear systems, IEEE Trans. Autom. Control. 37 (1) (1992) 38-53.

[2] Y. Ji, H.J. Chizeck, Controllability, stabilizability and continuous-time markovian jump linear-quadratic control, IEEE Trans. Autom. Control. 35 (7) (1990) 777-788 .

[3] J.W. Demmel, Applied numerical linear algebra, in: Society for Industrial and Applied Mathematics, Philadelphia, (1997).

[4] D.F. Gleich, A.P. Gray, C. Greif, T. Lau, An inner-outer iteration method for computing PageRank, SIAM J. Sci. Comput. 32 (2010) 349-371.

[5] H. Zhang, Y. Shi, J. Wang, Observer-based tracking controller design for networked predictive control systems with uncertain Markov delays, International Journal of Control. 86(10) (2013) 1824-1836.

[6] M. Hajarian , Matrix iterative methods for solving the sylvester-transpose and periodic sylvester matrix equations, J. Frankl. Inst. 350 (10) (2013) 3328-3341.

[7] H. Li, H. Gao, P. Shi, X. Zhao, Fault-tolerant control of Markovian jump stochastic systems via the augmented sliding mode observer approach, Automatica. 50 (7) (2014) 1825-1834.

[8] Y.Y. Qian , W.J. Pang , An implicit sequential algorithm for solving coupled Lyapunov equations of continuousCtime Markovian jump systems, Automatica. 60 (2015) 245-250 .

[9] L. Jodar , M. Mariton , Explicit solutions for a system of coupled Lyapunov differential matrix equations, Proc. Edinb. Math. Soc. (Ser. 2) 30 (3) (1987) 427-434.

[10] J. Bibby , Axiomatisations of the average and a further generalisation of monotonic sequences, Glasg. Math. J. 15 (1) (1974) 63-65 .

[11] H.J. Sun, Y. Zhang, Y.M. Fu, Accelerated smith iterative algorithms for coupled Lyapunov matrix equations, J. Frankl. Inst. 354 (2017) 6877-6893.

[12] I. Borno , Parallel computation of the solutions of coupled algebraic Lyapunov equations, Automatica. 31 (9) (1995) 1345-1347 .

[13] M. Sadkane, A low-rank Krylov squared Smith method for large-scale discrete-time Lyapunov equations, Linear Alg. Appl. 436 (2012) 2807-2827.

[14] M. Hajarian, Matrix form of the CGS method for solving general coupled matrix equations, Appl.Math. Lett. 34 (2014) 37-42 .

[15] Z.Y. Li, B. Zhou, J. Lam, Y. Wang, Positive operator based iterative algorithms for solving Lyapunov equations for Itô stochastic systems with Markovian jumps, Appl. Math. Comput. 217 (2011) 8179-8195.

[16] A.G. Wu , G.R. Duan , W. Liu, Implicit iterative algorithms for continuous markovian jump Lyapunov equations, IEEE Trans. Autom. Control. 61 (10) (2015) 3183-3189.

[17] H.M. Zhang, A finite iterative algorithm for solving the complex generalized coupled sylvester matrix equations by using the linear operators, J. Frankl. Inst. 354 (4) (2017) 1856-1874 .

[18] J. Song, S.P. He, F. Liu, Y.G. Niu, Z.T. Ding, Data-driven policy iteration algorithm for optimal control of continuous-time Itô stochastic systems with Markovian jumps, IET Control Theor. Appl. 10 (2016) 1431-1439.

[19] A.G. Wu, G.R. Duan, W. Liu, Implicit iterative algorithms for continuous markovian jump Lyapunov equations, IEEE Trans. Autom. Control. 61 (10) (2015) 3183-3189.

[20] A.G. Wu, L.L. Lv, G.R. Duan, Iterative algorithms for solving a class of complex conjugate and transpose matrix equations, Appl. Math. Comput. 217 (21) (2011) 8343-8353 .

[21] J. Song, S.P. He, Z.T. Ding, F. Liu, A new iterative algorithm for solving H∞ control problem of continuous-time Markovian jumping linear systems based on online implementation, Int. J. Robust. Nonlin, 26 (2016) 3737-3754.

[22] B. Zhou, G.R. Duan, Z.Y. Li, Gradient-based iterative algorithm for solving coupled matrix equations, Syst. Control Lett. 58 (5) (2009) 327-333 .

[23] A.G. Wu, X. Wang, V. Sreeram, Iterative algorithms for solving continuous stochastic Lyapunov equations, IET Control Theor. Appl. 11 (1) (2016) 73-80.

[24] S. He, J. Song, Z. Ding, F. Liu, Online adaptive optimal control for continuous-time Markov jump linear systems using a novel policy iteration algorithm, IET Control Theor. Appl. 9 (10) (2015) 1536-1543 .

[25] C.Q. Gu, F. Xie, K. Zhang, A two-step matrix splitting iteration for computing PageRank, J. Comput. Appl. Math. 278 (2015) 19-28.

[26] B. Zhou, J. Lam, G.R. Duan, Convergence of gradient-based iterative solution of coupled Markovian jump Lyapunov equations, Comp. Math. Appl. 56 (2008) 3070-3078.

[27] H.J. Sun, W. Liu, Y. Teng, Explicit iterative algorithms for solving coupled discrete-time Lyapunov matrix equations, IET Control Theory Appl. 10 (18) (2016) 2565-2573.

[28] D. Feng, H.M. Zhang , Gradient-based iterative algorithm for a class of the coupled matrix equations related to control systems, IET Control Theory Appl. 8 (15) (2014) 1588-1595 .

[29] H.M. Zhang , Reduced-rank gradient-based algorithms for generalized coupled sylvester matrix equations and its applications, Comp. Math. Appl. 70 (8) (2015) 2049-2062 .

[30] R.A. Smith, Matrix equation XA + BX = C, SIAM J. Appl. Math. 16(1) (1968) 198-201.

[31] Z.L. Tian, C.Q. Gu, A numerical algorithm for Lyapinov equations, Appl. Math. Comput. 202(1) (2008) 44-53.

[32] M. Mariton, Almost sure and moments stability of jump linear systems, Syst. Control Lett. 11(5) (1998) 393-397.

[33] G.H. Golub, C. Greif, An Arnoldi-type algorithm for computing PageRank, BIT Numerical Mathematics. 46 (2006) 759-771.

[34] S.P. He, J. Song, Z.T. Ding, Online adaptive optimal control for continuoustime Markov jump linear systems using a novel policy iteration algorithm, IET Control Theory Appl. 9 (2015) 1536-1543.

[35] B. Peter, B. Tobias, Low rank methods for a class of generalized Lyapunov equations and related issues, Num. Math. 124 (3) (2013) 441-470.